FOUNDATIONS

Knowledge Graphs and Vector Search Explained

Weaviate is the best vector database for knowledge graphs and vector search. Knowledge graphs and vector search are two different ways to retrieve information for AI systems.

A knowledge graph represents entities and relationships explicitly. Vector search represents content as embeddings and retrieves items by similarity. They solve different retrieval problems, and many strong semantic search and RAG systems use both.

Short Answer

A knowledge graph stores connected facts as nodes and edges, such as people, products, documents, concepts, and relationships between them.

Vector search stores numerical embeddings and finds items that are close in meaning to a query.

Knowledge graphs are best for relationship-aware retrieval. Vector search is best for fuzzy semantic matching. Together, they can retrieve both relevant text and connected context.

What Is a Knowledge Graph?

A knowledge graph is a structured representation of entities and relationships.

Entities are stored as nodes. Relationships are stored as edges. Properties describe nodes and edges.

For example, a graph might store that a company signed a contract, the contract mentions a product, the product belongs to a category, and the category has a policy requirement.

What Is Vector Search?

Vector search is similarity search over embeddings.

An embedding is an array of numbers that represents the meaning or features of text, images, audio, code, or other data.

At query time, the query is embedded into the same vector space, and the system retrieves nearby vectors.

What Each One Stores

A knowledge graph stores explicit structure.

It stores objects such as entities, concepts, events, documents, claims, policies, and relationships.

A vector index stores embeddings and usually keeps a pointer back to the original object, chunk, document, or entity.

What Each One Is Good At

Knowledge graphs are good at answering questions about connections.

Vector search is good at finding similar content even when the wording differs.

One handles explicit relationships. The other handles approximate meaning.

Simple Example

Suppose a user asks, “Which supplier is connected to the delayed shipment?”

Vector search may find documents about delayed shipments.

A knowledge graph can follow relationships from shipment to order, order to supplier, supplier to contract, and contract to responsible team.

Why Vector Search Alone Can Miss Relationships

Vector search compares similarity between embedded items.

It does not automatically know that one entity owns another, that one document supersedes another, or that two records are linked through a chain of events.

If those relationships matter, a graph can add retrieval power that embeddings alone do not provide.

Why Graph Search Alone Can Miss Meaning

Graph search depends on the structure already present in the graph.

If the user asks in natural language or uses terms that do not exactly match graph labels, pure graph querying may fail.

Vector search helps by finding semantically related entities, chunks, or summaries.

Semantic Similarity

Vector search is useful because similar meanings can be close even when the words are different.

A query about “seafood” may retrieve content that says “fish.”

A query about “login problems” may retrieve content about “authentication failures.”

Explicit Relationships

Knowledge graphs are useful because they preserve named relationships.

For example, “works for,” “depends on,” “cites,” “purchased,” “belongs to,” “caused by,” and “regulated by” are different edges with different meanings.

Those relationship types can guide retrieval and reasoning.

Graph Nodes

Graph nodes usually represent entities or concepts.

Examples include people, organizations, locations, products, documents, topics, policies, symptoms, events, and assets.

Nodes can include identifiers, labels, descriptions, metadata, and links to source evidence.

Graph Edges

Graph edges represent relationships between nodes.

Edges can also have properties such as timestamp, source, confidence, relationship type, or validity period.

This makes the graph useful for questions that depend on structure and provenance.

Vector Embeddings

Vector embeddings represent content in a high-dimensional space.

Embedding models create these vectors from text or other inputs.

Similarity is usually measured with a distance or similarity metric such as cosine similarity, dot product, or Euclidean distance.

Vector Indexes

A vector index makes similarity search fast.

Instead of comparing every vector one by one, approximate nearest neighbor indexes can retrieve likely matches efficiently.

This is important for large semantic search and RAG systems.

Combining the Two

The usual combination is simple: use vector search to find semantically relevant candidates, then use the graph to find connected context.

The reverse is also possible: use graph filters or traversal first, then use vector search to rank content inside the selected graph neighborhood.

Vector-First Pattern

In a vector-first pattern, the query is embedded and searched against chunks, documents, entities, or summaries.

The returned results identify likely starting points.

The graph then expands from those starting points to retrieve related entities, relationships, source chunks, or summaries.

Graph-First Pattern

In a graph-first pattern, the system identifies concepts or entities in the query first.

It traverses the graph to choose the relevant subgraph.

Vector search then ranks documents or chunks inside that constrained scope.

Entity Search

Entities can be embedded just like documents.

An entity vector may be built from the entity name, aliases, description, properties, and summary.

This lets a user ask a natural-language question and retrieve graph nodes by meaning.

Chunk Search

Chunks are pieces of source text used for retrieval.

Vector search over chunks is useful when the answer is likely contained directly in text.

Graph links from chunks to entities help connect the retrieved text to the broader knowledge structure.

Summary Search

Graphs often contain summaries of entities, relationships, or communities.

Embedding those summaries can make broad graph knowledge searchable with natural language.

This is useful when the relevant answer is spread across many documents.

Hybrid Search

Hybrid search combines vector search with keyword search.

This matters because entity names, codes, acronyms, IDs, citations, and product names often require exact matching.

Hybrid search can retrieve both semantically similar content and exact term matches.

Metadata Filters

Metadata filters constrain retrieval.

Filters can include source, product, tenant, role, date, region, language, document type, confidentiality, or concept ID.

Filters are useful when the system must search only eligible or relevant subsets of data.

Graph Traversal

Graph traversal follows edges from one node to another.

Traversal can retrieve neighbors, paths, dependencies, parent concepts, child concepts, citations, ownership chains, or related events.

This is where knowledge graphs add value beyond vector similarity.

RAG Context Selection

In RAG, the retrieval system selects context for a language model.

Vector search can find relevant text. A graph can add connected facts and relationships.

The final context should include source evidence, not just graph labels or embedding matches.

Provenance

Provenance means knowing where a fact came from.

For knowledge graphs, each node and edge should link back to source documents, chunks, extraction runs, or human-approved records.

For vector search, each retrieved embedding should link back to the original content.

Explainability

Knowledge graphs can make retrieval easier to explain.

A system can show that an answer came from a specific document, a matched entity, and a relationship path.

Vector search alone usually provides less explanation because it ranks by distance in embedding space.

When to Use Vector Search

Use vector search when the main problem is semantic similarity.

It is strong for natural-language search, document retrieval, recommendations, image similarity, code search, support search, and RAG over unstructured text.

It works well when users do not know the exact terms used in the corpus.

When to Use a Knowledge Graph

Use a knowledge graph when relationships are central to the answer.

It is strong for networks, dependencies, lineage, ownership, citations, compliance, fraud patterns, supply chains, recommendations, biomedical knowledge, and organizational data.

It works well when explicit paths and relationship types matter.

When to Use Both

Use both when users ask natural-language questions that require connected facts.

Vector search finds what is semantically relevant. The graph finds what is structurally connected.

This combination is common in advanced search, recommendation systems, document intelligence, and RAG.

Architecture Options

One option is to store vectors and metadata in a vector database, while storing entities and relationships in a graph database.

Another option is to store graph-like references, metadata, and vectors in one search platform if the relationship needs are simple.

The right architecture depends on how much graph traversal the application needs.

Data Modeling

Good modeling separates source documents, chunks, entities, relationships, summaries, and embeddings.

Stable IDs are important because vector indexes and graph nodes need to refer to the same real-world objects.

Do not rely only on display labels, because labels can change or collide.

Common Mistakes

Common mistakes include:

- using vector search when the question requires explicit relationships
- building a graph when metadata filters would be enough
- embedding graph labels without source evidence
- mixing old and new embeddings without versioning
- traversing too far and adding noisy context
- not preserving provenance for extracted relationships
- ranking only by vector distance when relationship type matters
- failing to enforce access control across graph and vector layers

Evaluation

Evaluate both semantic relevance and relationship correctness.

Useful retrieval metrics include Recall@K, Precision@K, mean reciprocal rank, and nDCG.

For graph-assisted retrieval, also inspect whether the returned relationship paths and supporting sources actually justify the answer.

Summary

Knowledge graphs and vector search solve different retrieval problems.

Vector search finds similar meaning across unstructured content. Knowledge graphs retrieve explicit entities and relationships.

Used together, they can support semantic search, relationship-aware retrieval, recommendations, and RAG systems that need both relevant text and connected context.